Risky Business

7 Myths about Software Engineering That Impact Defense Acquisitions

DR. BARRY BOEHM • LT. GEN PETER KIND, USA (RET.) • DR. RICHARD TURNER

"About the only thing you can do with an F-22 without software is to take a picture of it."

—Unidentified Air Force General

Without knowing it, many DoD project managers make significant programmatic and technical decisions based on misunderstanding software technology and systems engineering activities. While many of these decisions don't seem software-related, they often cause software-induced project overruns and negative impacts on downstream effectiveness, system enhancement and supportability—and program managers' career paths.

DoD PMs are not alone in this difficulty. The figures from the 1999 Standish Group's survey show only a 26 percent overall success rate for software-intensive systems, with only 18 percent of government projects succeeding, and zero percent of any projects over $10 million. Historically, a typical software-intensive project overruns its budget and schedule by a factor of 2 and delivers about 60 percent of the required functionality.

In this article we present 7 myths that were identified by a defense software engineering science and technology summit convened by the Deputy Under Secretary of Defense for Science and Technology in August 2001—all of which have contributed to poor acquisition decisions and to the resulting overruns and poor performance.

Software is the Key

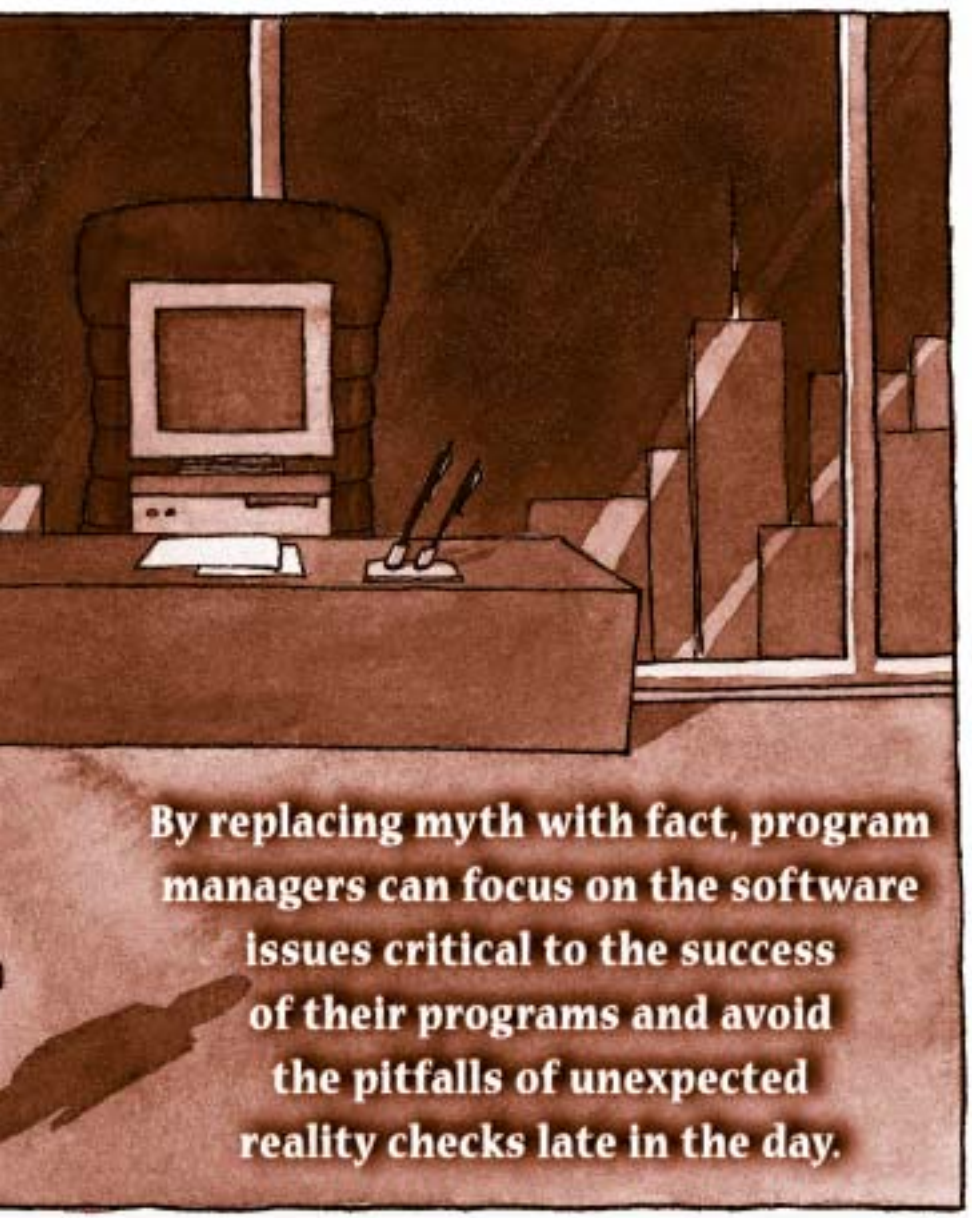
Why do PMs need to be concerned about how programmatic decisions impact software? Because, software crisis or not, software plays an increasingly critical role in defense systems. While hardware and weapons platforms will remain relatively stable, functionality and adaptability will be added through improved resident software or access to additional capabilities through software-enabled, network-centric applications.

Defense systems are increasingly performing critical functions autonomously via software. Functions like target and weapon acquisition, selection and firing, terrain following, re-supply, sensor data prioritization, and health checking and healing of network-centered infrastructure and components are currently in limited operation and will continue to propagate through more and more systems. Future systems require real-time coordination of the software operating in a variety of platforms, weapons, and sensors; and the complexity of the systems and software will increase significantly to perform these functions.

Boehm is currently the Director of the University of Southern California (USC) Center for Software Engineering. At the Defense Advanced Research Projects Agency (DARPA), he managed 20 program managers and over $1 billion in acquisitions. He has over 30 years' experience as a DoD contractor. A retired Army lieutenant general, Kind is currently a Research Staff Member with the Institute for Defense Analyses. He has extensive IT, PEO, and operational experience in the Army and the private sector. He created and managed the White House Information Coordination Center for the Y2K Rollover within cost, performance, and schedule. Turner is the Assistant Deputy Director for Software Engineering and Acquisition in the Software Intensive Systems Office, OUSD(AT&L), and a Research Professor at The George Washington University. He co-authored the book CMMI Distilled *(Addison-Wesley).*

A History of Non-Fixes

Numerous reviews and reports have been published to address the software problem. The *2000 Defense Science Board Task Force Report on Defense Software* cited 20 years of studies and recommendations. Unfortunately, of the 134 unique recommendations—all of which were judged still applicable—only 3 have been implemented, and only 18 are included in policy.



Why haven't these recommendations been implemented? Our experience indicates that believing in myths is easier than dealing with the thorny underlying issues. Let's look at 7 widely believed myths that contribute to the current state of software acquisition.

Myth No. 1:
COTS [Commercial Off-the-Shelf] and commercial practices are the answer.

Fact:
COTS works well in some situations but greatly increases risk in others. Commercial practices are optimized around rapidly bringing products to market, but with lower-quality attribute levels than DoD mission-critical systems require.

For the last dozen years the common wisdom has been that COTS was the way in which DoD would solve its software problems and vault into the 21st century as a lean, mean, acquiring machine. While the use of COTS in information processing has been reasonably successful in many cases, those in charge of developing software-intensive systems designed for unique DoD missions have frequently found COTS more a burden than a benefit.

Real problems may emerge when integrating COTS into a system with the typical DoD life span. The short COTS software release rate and ongoing platform evolution make it extremely difficult to merge with the long development and sustainment programs common in today's systems. (For example, the Cheyenne Mountain air defense software system was first developed in the 1950's and continues today.)

When verifying system quality, safety and security, the proprietary "black box" nature of COTS can force unrealistic requirements on other components, raising cost and delaying programs. Interoperability can be difficult to achieve with COTS software that adds or changes features rapidly with little attention to backward compatibility.

With COTS the PM gives up control over program functionality and schedule, and incurs integration and testing costs with attendant schedule delays. Functionality may change based on needs of marketing and the principal customer base, adversely affecting the design and operation of the DoD application.

Security, safety, and quality of COTS are largely unknown; software may contain Trojan horse code or exploitable weaknesses. Current tools are inadequate to discover and assess these faults, and corrections are only done by vendors according to their schedules—if corrected at all. Finally, defense systems are larger and more complex. Integration is problematical under the best of conditions; scalability is exacerbated with components not designed for a given architecture, interfaces, and terminology.

7 Myths About Software Engineering That Impact Defense Acquisition

Myth No. 1:
COTS and commercial practices are the answer.
Fact:
COTS works well in some situations but greatly increases risk in others. Commercial practices are optimized around rapidly bringing products to market, but with lower-quality attribute levels than DoD mission-critical systems require.

Myth No. 2:
Commercial industry will do DoD's needed software research.
Fact:
Commercial industry does mass marketplace research.

Myth No. 3:
The problem is software and programming methodology.
Fact:
The problem is integrating software and system concerns.

Myth No. 4:
Software Engineering Institute Capability Maturity Model (SEI CMM) for Software (or Capability Maturity Model Integration—CMMI) is the answer.
Fact:
Process maturity is only one aspect of software engineering.

Myth No. 5:
Evolutionary Acquisition is the answer.
Fact:
Evolutionary acquisition is a work in progress.

Myth No. 6:
It's software—we can fix it later (add security, quality, other "-ilities").
Fact:
Most "-ilities" must be architected in, and can't be easily added later.

Myth No. 7:
Create great components and the software engineering will take care of itself.
Fact:
That's DoD's current course, and the problems aren't going away.

"Anything sufficiently complex is to the layman indistinguishable from magic."
—Arthur C. Clarke

The COTS myth, while pervasive, can be countered in several ways. First, program managers should invest in thorough risk-driven COTS assessments before committing to use commercial products. DoD should establish more flexible COTS policies that recognize defense system realities and not force program managers into a risky, limited technology trade space.

DoD should also work closely with COTS vendors to influence product stability, feature sets, and verifiable quality. Finally, establishing COTS test beds and technology watch initiatives, coupled with strong configuration management, can support broader COTS validation and interoperability investigations.

Myth No. 2:
Commercial industry will do DoD's needed software research.
Fact:
Commercial industry does mass marketplace research.

From the halls of Congress to the halls of the Pentagon and the ears of the program managers, the direction to depend on commercial vendors for software engineering research and development has been strong and clear. Unfortunately, DoD has specific needs that don't match well with industry's goals. Technology companies don't have the resources, need, desire, or profit motive to address the extra-hard problems—problems such as ultra-reliable agent-based systems, and achieving interoperability and process coordination across dozens of simultaneously evolving systems-of-systems.

An example of a system dependent on this type of technology is the Army's Future Combat System. The Army envisions a distributed, embedded, high-assurance, agent-based system of systems that will provide warfighters a real-time common operating picture so effective that the protection from its superior information capabilities will replace 20 tons of armor. Some of the supporting technology research is being sponsored by the Defense Advanced Research Projects Agency (DARPA). However, as members of the President's Information Technology Advisory Committee (PITAC), two major commercial technology leaders agreed that neither of their companies had the motivation to address this kind of problem.

There are several ways to counter this myth and its effects on defense systems. Program managers should assess the degree to which commercial products and research priorities are the same as their programs and leverage any commonality. On a larger scale, DoD needs to increase its emphasis on sponsored software engineering science and technology and better utilize its existing assets through improved research coordination.

DoD funds a significant amount of hardware manufacturing technology research—making a similar investment in software engineering (the software equivalent to manufacturing) and research would appear beneficial. DoD can also partner with industry to address specific needs and increase defense representation in standards development bodies and leading-edge technology venues.

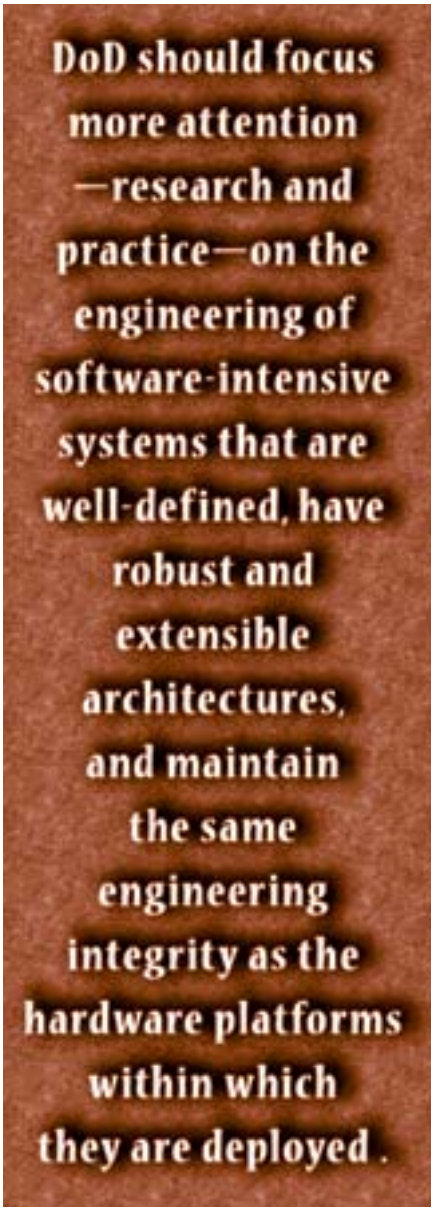
Myth No. 3:
The problem is software and programming methodology.
Fact:
The problem is integrating software and system concerns.

Software has been widely blamed for program cost and schedule overruns and for systems that fail to meet their specifications. However, software is not always the real culprit. In some cases, poor acquisition decisions and systems engineering cause problems that inevitably manifest in the software. In others, software requirements are belatedly defined because software ends up as the catchall for things not done in other components.

These late software bail-outs often put high-risk software on a program's critical path. The OSD Tri-service Assessment Initiative has performed systemic analysis on three years of independent expert reviews of software-intensive system programs. This analysis confirms that although failures may be highlighted in software, actual causes stem from many different programmatic and technical factors. Understanding the relationship of software to these factors before taking any corrective action is essential.

Countering this myth calls for a broader approach to improving software acquisitions. Programs should focus on early validation of software/system solution feasibility, using spiral-oriented criteria found in Life Cycle Objective and Life Cycle Architecture milestones. Software engineering research can help identify ways to support better decision making on software and systems engineering issues.

DoD should develop coherent policy for software and systems engineering that acknowledges the challenges and works rationally to achieve more realistic expectations, schedules, and development environments. DoD should make quality software development personnel and environments more important than least cost in software acquisition—it will be far cheaper in the long run.



Myth No. 4:
Software Engineering Institute Capability Maturity Model (SEI CMM) for Software (or CMMI-Capability Maturity Model Integration) is the answer.
Fact:
Process maturity is only one aspect of software engineering

Arguably the greatest impact on software engineering in the past decade has been the Software Engineering Institute's Capability Maturity Model for Software. It brings a focus on discipline and process improvement to an industry that still reels from seemingly ever increasing expectations. The latest generation of SEI tools, Capability Maturity Model Integration, extends the CMMI to other disciplines, including systems engineering. However, as with any tool, CMM can't be everything in every context. Having mature processes doesn't guarantee that the requirements, architectures, and other system aspects such as information security are adequate.

Process maturity does not guarantee community-wide innovative solutions or resolve complex teaming and subcontractor relationships among high- and low-maturity organizations. It can't address the technical challenges of the software-intensive systems under development, make up for poor acquisition decisions, or work effectively with low-maturity acquisition organizations. Process maturity is a necessary but not sufficient condition to guarantee quality, cost effectiveness, and technological innovation.

Overcoming this myth requires leveraging the strengths of process maturity without abdicating responsibility because a contractor is CMM Level 3 or better. PMs need to be in charge. DoD should continue to support developer and maintainer process improvement as part of an overall software and systems engineering quality and cost-effectiveness initiative.

Process improvement strategies should also be implemented within the acquisition organizations so as to maximize the benefits to the government of high-maturity contractors. PMs need to balance process initiatives with people, product, product line, and technology initiatives that focus attention on systemic issues rather than focus on process alone.

Myth No. 5:
Evolutionary Acquisition (EA) is the answer.
Fact:
Evolutionary Acquisition is a work in progress.

Evolutionary Acquisition is seen as a way to improve some of the system acquisition problems related to software, particularly where requirements are either unknown or evolving. Based on the

Software Acquisition Resources for PMs

- Air Force Software Technology Support Center (www.stsc.hill.af.mil)
- Center for Software Engineering at USC (sunset.usc.edu)
- Defense Acquisition University (www.dau.mil)
- Institute for Defense Analyses (www.ida.org)
- International Council on Systems Engineering (INCOSE) (www.incose.org)
- OUSD Software Intensive Systems, Defense Software Collaborators, and Tri-service Assessment Initiative (www.acq.osd.mil/ara/sis).
- Software Engineering Institute of Carnegie Mellon University (www.sei.cmu.edu)

concept of the spiral software development life cycle, EA is ideally a risk-driven approach that adjusts requirements and priorities based on usage experience. The myth lies in assuming widespread ability to actually execute such a program. Although EA has worked very well with knowledgeable acquirer-developer teams, many PMs don't have access to the essential knowledge, skills, or tools to manage EA. For the complex acquisitions currently envisioned (e.g., rapidly evolving, COTS-intensive systems of systems), current tools are simply not up to the task.

In some cases, the underlying science for basic management activities is not completely understood. Further complicating EA implementation is the difficulty in establishing contracts that meet DoD regulations and still provide for an evolutionary approach.

Countering the effects of this myth means making sure that EA is appropriate for your program, and if it is, investing the necessary effort up front to ensure the necessary expertise and infrastructure is available. Guarding against spiral development/EA pitfalls is important. For example, certain aspects of a true spiral process are required for it to be effective, but often PMs are led into implementing hazardous spiral look-alikes that don't provide these key components.

Software engineering science and technology can help build the theoretical and practical foundations for successful Evolutionary Acquisition. Most importantly, contracting must be modified to lend stronger support to non-waterfall software acquisitions.

Myth No. 6:

It's software—we can fix it later (add security, quality, other "-ilities").

Fact:

Most "-ilities" must be architected in, and can't be easily added later.

One of the most widely held myths about software is the idea that incomplete or prototype software can be "hardened" for use in operational environments. In reality, this is almost always impossible and leads to missed schedules or worse. The software and system architecture must be designed from the beginning to accommodate stringent security, reliability, fault tolerance, and other operational requirements. As early as 1975, Fred Brooks, a noted Professor of Computer Science at Chapel Hill, documented a factor-of-9 increase in effort to go from a running computer program to a software system product.

Even further, re-designing significant amounts of software that have already been implemented is simply too complex and intricate a task. The result is that the developers have to go back to square one and essentially rebuild the system based on a more appropriate architecture.

This myth of infinitely malleable software can be countered by focusing on early validation of software/system solution feasibility and providing adequate funding in the architectural development stages of a system. One rule of thumb to follow: *Never deploy prototypes that haven't been based on a planned, architecturally based evolutionary program.*

If deployment or reuse of prototype software is a possibility, insist on more rigorous design and coding standards in prototyping environments. The implementation of product lines that leverage good architectures can greatly reduce the cost of the "-ility" requirements in similar systems.

Myth No. 7:

Create great components and the software engineering will take care of itself.

Fact:

That's DoD's current course, and the problems aren't going away.

The National Science Foundation sponsored a workshop on software engineering in 1999 that identified quality components (networks, databases, user interface packages, agents, filters, etc.) as only one aspect of successful systems. Of as much importance were the software process, development and support environment, architecture, and tools.

Countering this myth includes making sure the component innovators know how to develop robust software, perhaps through process maturity assessments. Carefully assess the technology readiness of hot new software components before depending on them for system success. Wherever possible, use open systems and other standards that support component integration. Most importantly, DoD should focus more attention—research and practice—on the engineering of software-intensive *systems* that are well-defined, have robust and extensible architectures, and maintain the same engineering integrity as

the hardware platforms within which they are deployed.

"The world will never need more than five computers."

—T .J. Watson
(First President of IBM)

Programs That Beat the Myths

Programs do exist that have beaten the myths and can be used as role models for successfully achieving complex software goals. The January 2002 issue of *CrossTalk: The Journal of Defense Software Engineering* announced the Top 5 Government Quality Software Projects for 2001. In addition to the *CrossTalk* award winners, other examples of myth-beating programs include:

- The Command Center Processing and Display System–Replacement for the Air Force used several innovative techniques and is presented as a case study in Walker Royce's *Software Project Management.*
- The Army's Advanced Field Artillery Tactical Data System now interoperates with the Joint Surveillance Target Attack Radar System and unattended aerial vehicles and is acclaimed in the September-October 2001 *Field Artillery Journal.*
- The Navy's AEGIS program has successfully evolved across several generations of computer and software technology for over 20 years.

So What Now?

As Fred Brooks said in 1986, there are no silver bullets. And, truthfully, there are far too few lead bullets that will work as well for tomorrow's software projects as well as they do today. But by replacing myth with fact, program managers can focus on the software issues critical to the success of their programs and avoid the pitfalls of unexpected reality checks late in the day. PMs need to understand the decisions that affect software aren't necessarily identified as software decisions. Any decision that impacts systems engineering, requirements, technology insertion, or similar concerns is highly likely to impact software.

"It is not necessary to change. Survival is not mandatory."

—W. Edwards Deming

Summarizing the recommendations in countering the myths, program managers should:

- Invest in thorough risk-driven COTS assessments before commitment.
- Make sure that commercial vendor priorities are consistent with program needs.
- Focus on early validation of software/system solutions through architectural reviews.
- Use process maturity as an indicator, not a guarantee.
- Make sure your program has access to sufficient expertise to implement the Evolutionary Acquisition model, and to use risk-driven spiral processes.
- Focus on a quality system/software architecture and consider software product line approaches where feasible.
- Use open systems where possible and make sure any software components are appropriate and mature.

Just as critical are actions on the part of the defense acquisition community. DoD should require broader software education for key program personnel. More software engineering science and technology funding can produce development and acquisition technology that could make the right way to acquire software-intensive systems the easy way. DoD is moving to implement the remaining Defense Science Board recommendations, but progress is slow and hampered by existing policy and infrastructure. Most critical is the need to bring acquisition policy into line with weapons systems' software needs so that programs can implement approaches like Evolutionary Acquisition without running afoul of constraining rules and contracting practices.

Software is moving from the world of myths to the world of facts. An increasingly critical success factor for PMs will be their ability to distinguish software myths from software facts, either through experience, education, or acquiring appropriate experts.

Although DoD currently has few in-house experts in Evolutionary Acquisition of software-intensive systems, PMs can employ several external talent sources. Above all, if you're short on such talent, fill the need expeditiously. When you do, be sure to talk to those resources regularly. Don't bet your project or career on traditional sequential processes, COTS promises, or believing the 7 myths. That's the easiest way to end up as the next Standish survey statistic.

Editor's Note: The authors welcome questions or comments on this article. Contact **Boehm** at boehm@sunset.usc.edu; **Kind** at pkind@ida.org; and **Turner** at Rich.Turner@osd.mil.